A Prediction-Sampling-Based Multilayer-Structured Latent Factor Model for Accurate Representation to High-Dimensional and Sparse Data

Di Wu, *Member, IEEE*, Xin Luo, *Senior Member, IEEE*, Yi He, *Member, IEEE*, and MengChu Zhou, *Fellow, IEEE*

***Abstract*—Performing highly accurate representation learning on a high-dimensional and sparse (HiDS) matrix is of great significance in a big data-related application such as a recommender system. A latent factor (LF) model is one of the most efficient approaches to the HiDS matrix representation. However, an LF model's representation learning ability relies heavily on an HiDS matrix's known data density, which is extremely low due to numerous missing data entities. To address this issue, this work proposes a prediction-sampling-based multilayer-structured LF (PMLF) model with twofold ideas: 1) constructing a loosely connected multilayered LF architecture to increase the known data density of an input HiDS matrix by generating synthetic data layer by layer and 2) constraining this synthetic data generating process through a random prediction-sampling strategy and nonlinear activations to avoid overfitting. In the experiments, PMLF is compared with six state-of-the-art LF- and deep neural network (DNN)-based models on four HiDS matrices from industrial applications. The results demonstrate that PMLF outperforms its peers in well-balancing prediction accuracy and computational efficiency.**

***Index Terms*—Deep forest, deep learning, generalized multilayer structure, high-dimensional and sparse (HiDS) data, latent factor (LF) model, missing data estimation.**

I. Introduction

A MATRIX is widely adopted to describe the relationships between two types of entities in industrial applications [1], [2], [3]. Matrices from many big-data-related cases, such as recommender systems or social networks, are commonly high-dimensional and sparse (HiDS) simply because the relationships among numerous entities are unlikely to be fully observed in practice [2], [3]. Although they are extremely sparse, they contain valuable knowledge and patterns, e.g., users' potential preferences on items in recommender systems [4]. Hence, how to perform accurate representation learning to them becomes a hot yet thorny issue [5], [6], [7].

Manuscript received 1 October 2021; revised 10 April 2022; accepted 14 August 2022. Date of publication 9 September 2022; date of current version 1 March 2024. This work was supported by the National Natural Science Foundation of China under Grant 62176070, Grant 62002337, and Grant 61902370. *(Corresponding author: Xin Luo.)*

Di Wu and Xin Luo are with the College of Computer and Information Science, Southwest University, Chongqing 400715, China (e-mail: wudi.cigit@gmail.com; luoxin@swu.edu.cn).

Yi He is with the Department of Computer Science, Old Dominion University, Norfolk, VA 23529 USA (e-mail: yihe@cs.odu.edu).

MengChu Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: zhou@njit.edu).

Digital Object Identifier 10.1109/TNNLS.2022.3200009

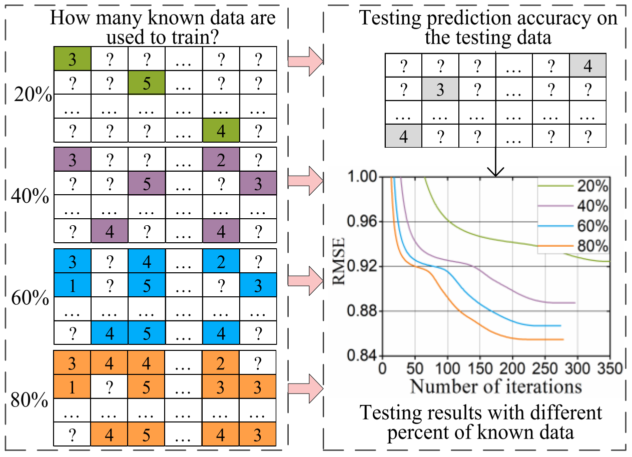
How many known data are used to train?
20%
40%
60%
80%
Testing prediction accuracy on the testing data
RMSE
Number of iterations
Testing results with different percent of known data
20%
40%
60%
80%

Fig. 1. Example: the known ratings of an HiDS matrix are extremely important for training an LF model.

Recently, a latent factor (LF) model has become one of the most successful and popular methods for this issue due to its high accuracy, simplicity, scalability, and flexibility [2], [3], [8]. Given an HiDS matrix, an LF model represents it by learning two low-rank LF matrices from its known data only [8]. The learned two matrices can be used to predict its missing data. From this point of view, an LF model's prediction accuracy relies heavily on a matrix's known data density, i.e., the number of known entries divided by the total number of entries in a matrix. An example in Fig. 1 is used to illustrate it.

Example 1: A typical HiDS matrix MovieLens 1M (https://grouplens.org/datasets/movielens/) from a recommender system is selected to illustrate a real case. It is randomly divided into fivefold, i.e., each fold contains 20% of its known data. Fourfolds are used for training, and the fifth one is used for testing. To simulate the different known data densities, we randomly select onefold (20%), twofold (40%), threefold (60%), and fourfold (80%) to train four different LF models under the same conditions. Then, they are evaluated on the testing set. Their prediction accuracies [root-mean-squared error (RMSE)] are shown in Fig. 1, where we observe that as the training data increase from onefold to fourfold, RMSE is reduced from 0.9244 to 0.8546, i.e., the prediction accuracy is improved by 7.55% [($\text{RMSE}_{\text{high}} - \text{RMSE}_{\text{low}}$)/$\text{RMSE}_{\text{high}}$].

Inspired by Example 1, we can collect more known data of a targeted HiDS matrix to train an LF model to improve its prediction accuracy. Such collection, however, is usually difficult in practice due to some limitations [2], [9]. For example, it is expensive, time-consuming, and sometimes impossible due to users' limited activities and thus limited real ratings in a recommender system. As a result, can we find an alternative way to increase the known data density of a targeted HiDS matrix for training an LF model?

Recently, Zhou and Feng [1] and Pang *et al.* [10] proposed deep forest to alleviate some drawbacks of deep neural networks (DNNs). Its principle is to gradually enhance a basic model by enriching its inputs via a carefully designed cascade multilayer learning structure, where each layer is an individual basic model, and its additional inputs come from the prediction outputs of its preceding layer. From this point of view, it becomes possible to gradually increase the known data density of a matrix to enhance an LF model by generating synthetic data via a carefully designed multilayer learning structure, where the generated synthetic data come from the prediction outputs of an LF model itself.

Motivated by Zhou and Feng [1] and Pang *et al.* [10], this study proposes a prediction-sampling-based multilayer-structured LF (PMLF) model to enhance an LF model's representation learning ability to an HiDS matrix. Its main idea is twofold: 1) a loosely connected multilayered LF structure is constructed to increase the known data density of an input matrix by generating synthetic data layer by layer and 2) such generating process is carefully constrained through a random prediction-sampling strategy and nonlinear activations to avoid overfitting. As such, PMLF has a multilayered cascade structure that is similar to deep forest [1], [10], i.e., PMLF's each layer is an individual LF model, and its additional input data come from the prediction outputs of its preceding LF model.

This study attempts to make the following contributions.

1) It first proposes to generate synthetic data to increase the known data density of an HiDS matrix to enhance an LF model's representation learning ability via a prediction-sampling-based multilayer structure.
2) It proposes a PMLF model with high accuracy and efficiency in predicting the missing data of an HiDS matrix.

Note that this study extends its previous conference version [52] and has the following additional contributions.

1) It improves the previous PMLF model's prediction accuracy by employing a weighting strategy to actively control the effects of generated synthetic data in representing an HiDS matrix.
2) It improves the previous PMLF model's computational efficiency by applying alternating stochastic gradient descent (ASGD) to achieve parallel training.
3) It theoretically analyzes the improved PMLF (named PMLF+) model, including convergence, algorithm design, and time complexity.
4) It conducts detailed and extensive empirical studies to evaluate PMLF+.

The results on four HiDS matrices from industrial applications demonstrate that PMLF+ outperforms its previous version PMLF and six state-of-the-art LF- and DNN-based models in well-balancing prediction accuracy and computational efficiency.

Section II introduces the related work. Section III states preliminaries. Section IV presents a PMLF+ model. Section V reveals experimental results. Section VI gives some discussions. Finally, Section VII concludes this article.

II. Related Work

After an LF model achieves great success in the high-profile competition of Netflix Prize [8], it has attracted much attention for analyzing HiDS data in the past decade. For example, it has become one of the most popular and successful approaches for implementing a recommender system [19], [37]. Besides, it has been widely applied to many big-data-related fields, such as data recovery in wireless sensor networks [51], services selection in online Web services [15], disease prevention in protein-protein networks [62], and traffic flow forecasting in intelligent transportation [63]. In addition, the LF model also has some potentials in feature selection [64], [65], [66], [67].

Various improved and sophisticated LF-based models have been proposed, including dual-regularization-based [12], bias-based [8], joint recommendation [14], probabilistic [13], neighborhood-and-location integrated [15], graph regularized [16], nonnegativity-constrained [2], confidence-driven [17], and data-characteristic-aware [37] ones. Although these models have different model designs, none of them constructs a deep structure to improve an LF model. Recently, our prior work [11] proposed a deep LF (DLF) model to gradually enhance its generalization ability by sequentially aggregating a series of LF models. Sharply different from DLF, this work constructs a prediction-sampling-based multilayer structure following the principle of deep forest [1]. With such a structure, we can gradually increase the known data density of an HiDS matrix to enhance an LF model's representation learning ability. As a result, when representing an HiDS matrix, our new model performs better than DLF.

Recently, DNNs attract much attention in processing HiDS data due to their powerful representation learning ability [18], [19], especially in recommender systems [20], [21], [22], [23], [24]. Zhang *et al.* [19] conducted a detailed review regarding this issue. Many DNN-based models [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35] have been proposed. Representative models include autoencoder-based [24], stacked autoencoder-based [26], denoising autoencoder-based [25], collaborative denoising autoencoder-based [28], hybrid autoencoder-based [27], neural collaborative filtering-based [20], recurrent neural network-based [29], hybrid deep structure-based [30], deep-matrix-factorization-with-neural-network-based [21], neural factorization-based [32],

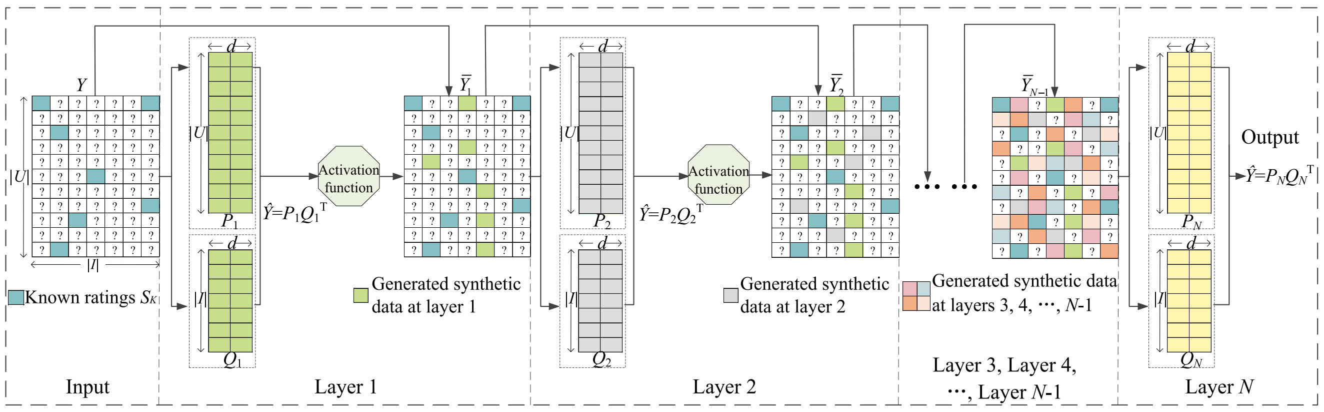

Fig. 2. Structure of flowchart a PMLF+ model.

multitask learning-oriented [31], attentional factorization-based [33], convolutional-matrix-factorization based [35], and deep-cooperative neural-network-based [34] ones.

Note that when representing an HiDS matrix, a DNN-based model's performance is achieved at the cost of high computation burden as they input the complete data rather than the known data of an HiDS matrix only [11]. For instance, dataset Epinion only has a known data density of 0.02%, where 13 668 321 ratings scatter in 120 492 columns and 755 760 rows [36]. If we input the complete data into a DNN-based model, more than 91 billion entries must be processed, which is both hugely time-consuming and computational resource-consuming. Besides, although a DNN-based model shows some promising advances in representing an HiDS matrix, Rendle *et al.* [53] demonstrated that an LF model still has competitive performance. In comparison, the proposed model is significantly different from a DNN-based one as follows: 1) it adopts an LF model as its fundamental component to construct its multilayer structure and 2) it relies only on the known data only rather than the complete data of an HiDS matrix. As a result, it achieves a much higher computational efficiency and lower memory requirement than DNN-based models, with highly competitive prediction accuracy.

III. Preliminaries

Definition 1 (An HiDS Matrix): Given two types of entity sets U and I, Y is a $|U| \times |I|$ matrix where each entry $y_{u,i}$ denotes the relationship between entity u ($u \in U$) and entity i ($i \in I$). K and K′, respectively, denote the sets of observed and unobserved entries of Y. Y is an HiDS matrix with $|\mathrm{K}| \ll |\mathrm{K}'|$ and large U and I.

Definition 2 (An LF Model): Given Y, an LF model is to learn two low-rank LF matrices P of size $|U| \times d$ and Q of size $|I| \times d$ to make Y's rank-d approximation $\hat{Y}$ by minimizing the distances between Y and $\hat{Y}$ on K only, where d is the LF dimension with $d \ll \min\{|U|, |I|\}$ and $\hat{Y} = PQ^{\mathrm{T}}$ can be obtained. $\hat{Y}$'s each entry $\hat{y}_{u,i}$ is the prediction for $y_{u,i}$.

From Definition 2, we see that an LF model works by designing an objective function to minimize the distances between Y and $\hat{Y}$ on K. Commonly, the Euclidean distance is adopted as a distance metric to design such function [8], i.e.,

$$\min_{P,Q}\varepsilon(P,Q) = \min_{P,Q}\frac{1}{2}\left\|\Omega \odot \left(Y - \hat{Y}\right)\right\|_F^2 = \min_{P,Q}\frac{1}{2}\left\|\Omega \odot \left(Y - PQ^{\mathrm{T}}\right)\right\|_F^2 \quad (1)$$

where $||\cdot||_F$ indicates the Frobenius norm of a matrix, $\odot$ indicates the Hadamard product (componentwise multiplication), and Ω indicates *a* $|U| \times |I|$ binary index matrix as follows:

$$\Omega_{u,i} = \begin{cases} 1, & \text{if } y_{u,i} \text{ is observed} \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

Note that (1) is an ill-posed problem because of overfitting. Hence, we incorporate Tikhonov regularization [2] into (1)

$$\min_{P,Q}\varepsilon(P,Q) = \min_{P,Q}\frac{1}{2}\left\|\Omega \odot \left(Y - PQ^{\mathrm{T}}\right)\right\|_F^2 + \frac{\lambda}{2}\left(\|P\|_F^2 + \|Q\|_F^2\right) \quad (3)$$

where λ is a regularization hyperparameter. In general, (3) can be solved by an optimization algorithm such as stochastic gradient descent (SGD) [60], [61].

IV. Proposed PMLF+ Model

A. Structure of PMLF+

Following the example shown in Fig. 1 and the principle of deep forest [1], the proposed PMLF+ model is designed, as shown in Fig. 2. It sequentially connects N basic LF (BLF) models and $N - 1$ nonlinear activation functions to construct its prediction-sampling-based multilayer structure. It works as follows:

1) inputting K into PMLF+ as the initial inputs;
2) initializing $n = 1$;
3) training LF matrices P_n and Q_n based on K and Λ only in parallel by noting that Λ is empty when $n = 1$;
4) sequentially and randomly selecting a missing entry $\langle u, i\rangle$ (an HiDS matrix's uth row and ith column entry) between two known entries in each row of $\overline{Y}_{n-1}$ as a blank entry by noting that the known data of $\overline{Y}_{n-1}$ consist of all the elements of Λ and K and $\overline{Y}_{n-1} = Y$ when $n = 1$;

5) predicting the blank entry based on P_n and Q_n to get $\hat{y}_{u,i}$;
6) inputting $\hat{y}_{u,i}$ into the activation function to reset its value to get $\overline{y}_{u,i}$, i.e., $\overline{y}_{u,i}$ is the output of the activation function;
7) putting $\overline{y}_{u,i}$ into Λ;
8) repeating Steps 4–7 until the generated synthetic data (the predictions generated by Steps 4–7, they are treated as known data for the next stage training) reach the preset threshold (i.e., the number of generated synthetic data equals the number of all known data);
9) $n = n+ 1$;
10) repeating Steps 3–9 until $n = N$;
11) output the learned LF matrices P_n and Q_N, which can be used to make Y's final approximation, i.e., $\hat{Y} = P_N Q_N^{\mathrm{T}}$.

Here, P_n and Q_n are the two LF matrices at the nth layer, $n \in \{1, \ldots, N\}$, N is a preset maximum number of layers, Λ is a set that contains all the synthetic data generated by all the previous $n-$ 1 layers at the nth layer, $\overline{Y}_{n-1}$ is the input HiDS matrix of the nth layer, and $\overline{y}_{u,i}$ is the output of the activation function corresponding to $\hat{Y}_{u,i}$.

B. Training the Nth Layer With SGD

To explain how to train a PMLF+ model, we describe the nth layer training process as a general case, $n \in \{1, \ldots, N\}$. We first extend (3) into the following single-LF-dependent form on a single element:

$$\min_{P_n,Q_n} \varepsilon(P_n, Q_n) = \min_{P_n,Q_n} \frac{1}{2} \sum_{\langle u,i\rangle \in \mathrm{K}} \left(y_{u,i} - \sum_{\tau=1}^{d} p_{u,\tau}^{n} q_{i,\tau}^{n} \right)^2 + \frac{\lambda}{2}\left(\|P_n\|_F^2 + \|Q_n\|_F^2 \right) \tag{4}$$

where $\langle u, i\rangle$ denotes an HiDS matrix's uth row and ith column entry, $p_{u,\tau}^{n}$ denotes P_n's uth row and τth column entry, and $q_{i,\tau}^{n}$ denotes Q_n's ith row and τth column entry . To input the synthetic data generated by all the previous $n-$ 1 layers into the nth layer, we transform (4) into the following form:

$$\begin{aligned} &\min_{P_n,Q_n} \varepsilon(P_n,\ Q_n) \\ &= \min_{P_n,Q_n} \underbrace{\frac{1}{2} \sum_{\langle u,i\rangle \in \mathrm{K}} \left(y_{u,i} - \sum_{\tau=1}^{d} p_{u,\tau}^{n} q_{i,\tau}^{n} \right)^2}_{\text{Training on input known data of HiDS matrix}} \\ &\quad + \underbrace{\frac{1}{2}\alpha \sum_{\langle u,i\rangle \in \Lambda} \left(\bar{y}_{u,i} - \sum_{\tau=1}^{d} p_{u,\tau}^{n} q_{i,\tau}^{n} \right)^2}_{\text{Training on generated synthetic data}} + \frac{\lambda}{2}\left(\|P_n\|_F^2 + \|Q_n\|_F^2 \right) \end{aligned} \tag{5}$$

where α is a coefficient of controlling the impact of synthetic data in Λ and $\overline{y}_{u,i}$ is an element of Λ. Note that when $n = 1$, Λ is empty because there is no generated synthetic data and PMLF+ degrades to a BLF model. Then, we consider the instant loss $\varepsilon_{u,i}^{n}$ of (5) on a single element $\forall\langle u, i\rangle \in \Lambda$ or K as follows:

$$\begin{cases} \text{if}\langle u, i\rangle \in \mathrm{K} : \varepsilon_{u,i}^{n} = \frac{1}{2}\left(y_{u,i} - \sum_{\tau=1}^{d} p_{u,\tau}^{n} q_{i,\tau}^{n} \right)^2 \\ +\frac{\lambda}{2}\left(\sum_{\tau=1}^{d} \left(p_{u,\tau}^{n}\right)^2 + \sum_{\tau=1}^{d} \left(q_{i,\tau}^{n}\right)^2 \right) \\ \text{if}\langle u, i\rangle \in \Lambda : \varepsilon_{u,i}^{n} = \frac{1}{2}\alpha\left(\bar{y}_{u,i} - \sum_{\tau=1}^{d} p_{u,\tau}^{n} q_{i,\tau}^{n} \right)^2 \\ +\frac{\lambda}{2}\left(\sum_{\tau=1}^{d} \left(p_{u,\tau}^{n}\right)^2 + \sum_{\tau=1}^{d} \left(q_{i,\tau}^{n}\right)^2 \right). \end{cases} \tag{6}$$

Next, if we adopt SGD to minimize (6) with respect to each single element, we have

$$\forall\tau \in \{1, 2, \ldots, d\}, \quad \forall\langle u, i\rangle \in \mathrm{K} \text{ or } \Lambda,\ y_{u,i} : \begin{cases} p_{u,\tau}^{n} \leftarrow p_{u,\tau}^{n} - \eta \frac{\partial \varepsilon_{u,i}^{n}}{\partial p_{u,\tau}^{n}} \\ q_{i,\tau}^{n} \leftarrow q_{i,\tau}^{n} - \eta \frac{\partial \varepsilon_{u,i}^{n}}{q_{i,\tau}^{n}} \end{cases} \tag{7}$$

where η is the learning rate. By extending (7), we have the training rules (8), as shown at the bottom of the page. After each element in K and Λ is trained with (8), P_n and Q_n are learned and can be used to generate synthetic data. Note that Λ is empty when $n = 1$. Hence, P_1 and Q_1 are only learned on K.

C. Generating Synthetic Data at the Nth Layer

Given an HiDS matrix, increasing its known data density is crucial for improving an LF model's representation learning ability. Following this principle, PMLF+ increases the data density of an HiDS matrix by predicting its missing data layer by layer. Let $\overline{Y}_{n-1}$ be the input HiDS matrix of the nth layer, where the known data of $\overline{Y}_{n-1}$ contains all the elements of Λ and K. The specific method of generating synthetic data at the nth layer is as follows:

1) among $\overline{Y}_{n-1}$, sequentially and randomly selecting a missing entry $\langle u, i\rangle$ between two known entries in each row of $\overline{Y}_{n-1}$ as a blank entry;
2) predicting the blank entry $\langle u, i\rangle$ based on P_n and Q_n, i.e., $\overline{y}_{u,i} = \sum d\tau = 1 p_{u,\tau}^{n} q_{i,\tau}^{n}$;
3) inputting $\overline{Y}_{u,i}$ into the activation function (9) to reset its value to get $\overline{y}_{u,I}$;
4) putting the output $\overline{y}_{u,i}$ of the activation function (9) into Λ;
5) repeating Steps 1–4 until the generated synthetic data reach the preset threshold (i.e., the number of generated synthetic data equals the number of all known data).

$$\forall\tau \in \{1, 2, \ldots, d\}, \begin{cases} \text{if}\langle u, i\rangle \in \mathrm{K} : \begin{cases} p_{u,\tau}^{n} \leftarrow p_{u,\tau}^{n} + \eta q_{i,\tau}^{n}\left(y_{u,i} - \sum_{\tau=1}^{d} p_{u,\tau}^{n} q_{i,\tau}^{n} \right) - \eta\lambda p_{u,\tau}^{n} \\ q_{i,\tau}^{n} \leftarrow q_{i,\tau}^{n} + \eta p_{u,\tau}^{n}\left(y_{u,i} - \sum_{\tau=1}^{d} p_{u,\tau}^{n} q_{i,\tau}^{n} \right) - \eta\lambda q_{i,\tau}^{n} \end{cases} \\ \text{if}\langle u, i\rangle \in \Lambda : \begin{cases} p_{u,\tau}^{n} \leftarrow p_{u,\tau}^{n} + \eta q_{i,\tau}^{n}\alpha\left(\bar{y}_{u,i} - \sum_{\tau=1}^{d} p_{u,\tau}^{n} q_{i,\tau}^{n} \right) - \eta\lambda p_{u,\tau}^{n} \\ q_{i,\tau}^{n} \leftarrow q_{i,\tau}^{n} + \eta p_{u,\tau}^{n}\alpha\left(\bar{y}_{u,i} - \sum_{\tau=1}^{d} p_{u,\tau}^{n} q_{i,\tau}^{n} \right) - \eta\lambda q_{i,\tau}^{n} \end{cases} \end{cases} \tag{8}$$

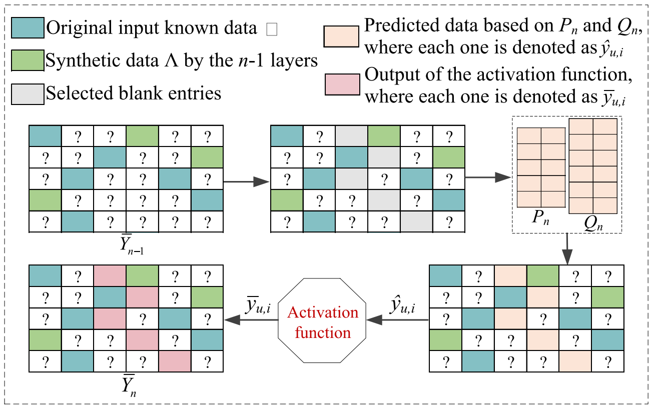
Original input known data
Synthetic data Λ by the n-1 layers
Selected blank entries
Predicted data based on P_n and Q_n, where each one is denoted as $\hat{y}_{u,i}$
Output of the activation function, where each one is denoted as $\bar{y}_{u,i}$
$\bar{Y}_{n-1}$
P_n
Q_n
$\hat{y}_{u,i}$
Activation function
$\bar{y}_{u,i}$
$\bar{Y}_n$

Fig. 3. Example of how to generate synthetic data at the nth layer.

The activation function is designed as follows:

$$\bar{y}_{u,i} = \begin{cases} y_{\min}+1/\left(1+e^{-\hat{y}_{u,i}}\right), & \text{if } \hat{y}_{u,i} < y_{\min} \\ y_{\max}/\left(1+e^{-\hat{y}_{u,i}}\right), & \text{if } y_{u,i} > y_{\max} \\ \hat{y}_{u,i}, & \text{otherswise} \end{cases} \quad (9)$$

where $y_{\min}$ and $y_{\max}$ denote the minimum and maximum value of Y, respectively. Obviously, $\hat{y}_{u,i} < y_{\min}$ or $\hat{y}_{u,i} > y_{\max}$ is incorrect. Equation (9) has the function of resetting the extremely unreasonable predictions. To illustrate how to generate synthetic data at the nth layer, an example is given in Fig. 3. First, we select the blank entries from $\overline{Y}_{n-1}$ randomly. Second, we predict data for the selected blank entries with P_n and Q_n. Finally, we perform the predictions via the activation function.

D. Parallelizing the Nth Layer With ASGD

From (8), we see that the two LF matrices of each layer of PMLF+ are trained iteratively. The processes of training P_n and Q_n are dependent on each other. According to [38], [39], and [40], an SGD-based LF model can be parallelized by relaxing the dependent training processes between P_n and Q_n. To this end, ASGD [40], [54] is employed to train each layer of PMLF+, i.e., alternatively fixing one of P_n and Q_n to train the other one during each training epoch. Fig. 4 shows how to train P_n in parallel by fixing Q_n at one training epoch. First, the input $\overline{Y}_{n-1}$ is separated into $|U|$ rows. Then, each separate row of $\overline{Y}_{n-1}$ is adopted to train the corresponding row of P_n only. As such, P_n can be trained in parallel with maximum $|U|$ blocks . Hence, based on (8), the training rules of P_n in parallel with $|U|$ blocks at one training epoch (10), as shown at the bottom of the next page, where $p^n_{u,\cdot}$ and $p^n_{i,\cdot}$ denote P_n's uth row and Q_n's ith row, respectively. For the sake of readability, we simplify the single-LF-dependent form of (8) into the vector-LF-dependent form of (10). After P_n is trained in parallel on all the known data of $\overline{Y}_{n-1}$, then we train Q_n in parallel by fixing P_n at the same training epoch. By analogy, we have the training rules of Q_n in parallel with $|I|$ blocks at one training epoch (11), as shown at the bottom of the next page.

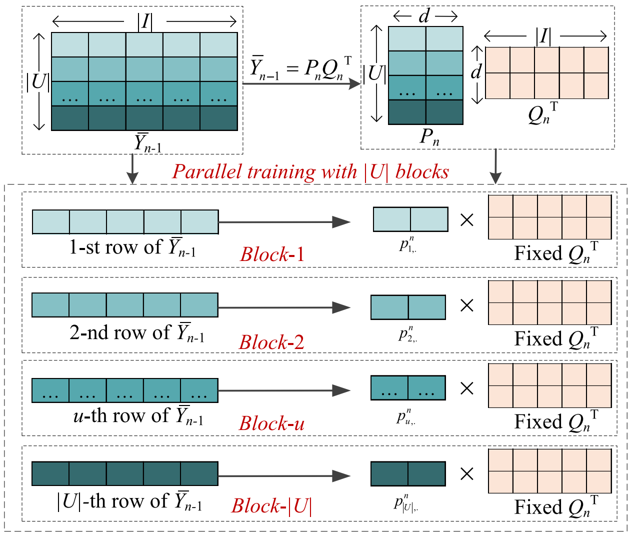
$|I|$
$|U|$
$\bar{Y}_{n-1}$
$\bar{Y}_{n-1}=P_nQ_n^{\mathrm{T}}$
d
P_n
Q_n^{T}
Parallel training with |U| blocks
1-st row of $\bar{Y}_{n-1}$
Block-1
$p^n_{1,\cdot}$
Fixed Q_n^{T}
2-nd row of $\bar{Y}_{n-1}$
Block-2
$p^n_{2,\cdot}$
u-th row of $\bar{Y}_{n-1}$
Block-u
$p^n_{u,\cdot}$
|U|-th row of $\bar{Y}_{n-1}$
Block-|U|
$p^n_{|U|,\cdot}$

Fig. 4. Parallelizing each training epoch of each layer of PMLF+: fixing Q_n to parallel train P_n.

E. Convergence Analysis and Algorithm Design

Theorem 1: Given an HiDS matrix Y, a PMLF+ model is convergent in training two LF matrices to make its rank-d approximation $\hat{Y}$ by minimizing the distances between Y and $\hat{Y}$ only on its observed entries set K with ASGD.

Proof: The proof of Theorem 1 is given in the Appendix.

Besides, we design Algorithm PMLF+ in Table I based on the above analyses. According to [2], [3], and [11], the time complexity of a BLF model is O(|K| $\times d \times$ $T_{\max}$), where $T_{\max}$ is the maximum iteration count. Section IV-B shows that each layer of PMLF is a BLF model. Thus, the time complexity of the 1st layer of PMLF+ is also O(|K| $\times d \times$ $T_{\max}$). Except for the 1st layer, some synthetic data are generated to help train. Suppose that the generated ratio of synthetic data is ν at each layer, i.e., ν = | generated synthetic data |/(|Λ|+ |K|). Then, the time complexity of the nth layer is O(|K| $\times d \times T_{\max} \times (1+\nu)^{n-1}$). If PMLF+ is composed of N layers, by summing the time complexity of each layer, the total time complexity of Algorithm PMLF+ is O(|K| $\times d \times T_{\max} \times (1+\nu)^N/\nu$). From Section IV-D, we see that PMLF+ can be trained in parallel with ASGD. Hence, Supposing that PMLF+ is trained in parallel with M number of threads, its final time complexity is O(|K| $\times d \times T_{\max} \times (1+\nu)^N/(\nu \times M)$). In terms of space complexity, PMLF+ adopts the following data structures: 1) an array with length |K|$\times (1+\nu)^{n-1}$ to cache the input data for training; 2) a matrix with size $|U| \times d$ to cache P_n; and 3) a matrix with size |I| $\times d$ to cache Q_n. Then, the space complexity of Algorithm PMLF+ is $\Theta(d\times \max\{$|K|$\times (1+\nu)^N/(\nu \times d)$, |U|, |I|$\})$.

V. Experiments

In the experiments, we aim at answering the following research questions (RQs).

RQ. 1: Does PMLF+ outperform its previous version PMLF and state-of-the-art models in representing an HiDS matrix?

RQ. 2: How does the prediction-sampling-based multilayer structure impact PMLF+'s performance?

RQ. 3: Are the generated synthetic data helpful for improving PMLF+ in representing an HiDS matrix?

RQ. 4: How does the LF dimension d impact PMLF+'s performance?

A. General Settings

1) Datasets: Four benchmark HiDS matrices from industrial applications [36], [41] are adopted in the experiments. We randomly select each matrix's 20% known data as the training set, and the remaining 80% ones are treated as the testing set. They are briefly described as follows.

1) *D1 (The Dating Dataset)*: It is collected from an online dating website LibimSeTi [42]. It has 135 359 rows and 168 791 columns with 17 359 346 known entries, which indicates that its percentage of known data is 0.08%.
2) *D2 (The Epinion Dataset)*: It is collected from the Trustlet website [36]. It has 755 760 rows and 120 492 columns with 13 668 321 known entries, which indicates that its percentage of known data is 0.02%.
3) *D3 (The Flixter Dataset)*: It is collected from the Flixter website [43]. It has 147 612 rows and 48 794 columns with 8 196 077 known entries, which indicates that its percentage of known data is 0.11%.
4) *D4 (The MovieLens 1M Dataset)*: It is collected from MovieLens system [44]. It has 6040 rows and 3706 columns with 1 000 209 known entries, which indicates that its percentage of known data is 4.47%.

2) Evaluation Protocol: In many big-data-related applications, it is highly significant to recover the unknown relationships among an HiDS matrix. Hence, the task of missing data prediction is widely adopted as the evaluation protocol of representing an HiDS matrix [19]. To evaluate prediction accuracy, RMSE and mean absolute error (MAE) are usually adopted [19]. They are computed as follows:

$$\mathrm{RMSE}=\sqrt{\left(\sum_{\langle u,i\rangle\in\Gamma}\left(y_{u,i}-\hat{y}_{u,i}\right)^2\right)\Big/|\Gamma|}$$

$$\mathrm{MAE}=\left(\sum_{\langle u,i\rangle\in\Gamma}\left|y_{u,i}-\hat{y}_{u,i}\right|_{\mathrm{abs}}\right)\Big/|\Gamma|$$

where $|\cdot|_{\mathrm{abs}}$ indicates the absolute value and Γ indicates the testing set. Besides, we test CPU running time to evaluate computational efficiency.

$$\begin{cases}\text{On } \bar{Y}_{n-1}, \text{ fixing } Q_n, \forall i\in\{1,2,\ldots,|I|\}, \text{ for each}\\ \langle 1,i\rangle\in \mathrm{K} \text{ or } \Lambda: \begin{cases} p^n_{1,.}\leftarrow p^n_{1,.}+\eta q^n_{i,.}\left(y_{1,i}-p^n_{1,.}q^n_{i,.}\right)-\eta\lambda p^n_{1,.}, & \text{if } \langle 1,i\rangle\in\mathrm{K}\\ p^n_{1,.}\leftarrow p^n_{1,.}+\eta q^n_{i,.}\alpha\left(\bar{y}_{1,i}-p^n_{1,.}q^n_{i,.}\right)-\eta\lambda p^n_{1,.}, & \text{if } \langle 1,i\rangle\in\Lambda\end{cases}\\ \langle 2,i\rangle\in \mathrm{K} \text{ or } \Lambda: \begin{cases} p^n_{2,.}\leftarrow p^n_{2,.}+\eta q^n_{i,.}\left(y_{2,i}-p^n_{2,.}q^n_{i,.}\right)-\eta\lambda p^n_{2,.}, & \text{if } \langle 2,i\rangle\in\mathrm{K}\\ p^n_{2,.}\leftarrow p^n_{2,.}+\eta q^n_{i,.}\alpha\left(\bar{y}_{2,i}-p^n_{2,.}q^n_{i,.}\right)-\eta\lambda p^n_{2,.}, & \text{if } \langle 2,i\rangle\in\Lambda\end{cases}\\ \ldots\ldots\\ \langle u,i\rangle\in \mathrm{K} \text{ or } \Lambda: \begin{cases} p^n_{u,.}\leftarrow p^n_{u,.}+\eta q^n_{i,.}\left(y_{u,i}-p^n_{u,.}q^n_{i,.}\right)-\eta\lambda p^n_{u,.}, & \text{if } \langle u,i\rangle\in\mathrm{K}\\ p^n_{u,.}\leftarrow p^n_{u,.}+\eta q^n_{i,.}\alpha\left(\bar{y}_{u,i}-p^n_{u,.}q^n_{i,.}\right)-\eta\lambda p^n_{u,.}, & \text{if } \langle u,i\rangle\in\Lambda\end{cases}\\ \ldots\ldots\\ \langle |U|,i\rangle\in \mathrm{K} \text{ or } \Lambda: \begin{cases} p^n_{|U|,.}\leftarrow p^n_{|U|,.}+\eta q^n_{i,.}\left(y_{|U|,i}-p^n_{|U|,.}q^n_{i,.}\right)-\eta\lambda p^n_{|U|,.}, & \text{if } \langle |U|,i\rangle\in\mathrm{K}\\ p^n_{|U|,.}\leftarrow p^n_{|U|,.}+\eta q^n_{i,.}\alpha\left(\bar{y}_{|U|,i}-p^n_{|U|,.}q^n_{i,.}\right)-\eta\lambda p^n_{|U|,.}, & \text{if } \langle |U|,i\rangle\in\Lambda\end{cases}\end{cases} \tag{10}$$

$$\begin{cases}\text{On } \bar{Y}_{n-1}, \text{ fixing } P_n, \forall u\in\{1,2,\ldots,|U|\}, \text{ for each}\\ \langle u,1\rangle\in \mathrm{K} \text{ or } \Lambda: \begin{cases} q^n_{1,.}\leftarrow q^n_{1,.}+\eta p^n_{u,.}\left(y_{u,1}-p^n_{u,.}q^n_{1,.}\right)-\eta\lambda q^n_{1,.}, & \text{if } \langle u,1\rangle\in\mathrm{K}\\ q^n_{1,.}\leftarrow q^n_{1,.}+\eta p^n_{u,.}\alpha\left(\bar{y}_{u,1}-p^n_{u,.}q^n_{1,.}\right)-\eta\lambda q^n_{1,.}, & \text{if } \langle u,1\rangle\in\Lambda\end{cases}\\ \langle u,2\rangle\in \mathrm{K} \text{ or } \Lambda: \begin{cases} q^n_{2,.}\leftarrow q^n_{2,.}+\eta p^n_{u,.}\left(y_{u,2}-p^n_{u,.}q^n_{2,.}\right)-\eta\lambda q^n_{2,.}, & \text{if } \langle u,2\rangle\in\mathrm{K}\\ q^n_{2,.}\leftarrow q^n_{2,.}+\eta p^n_{u,.}\alpha\left(\bar{y}_{u,2}-p^n_{u,.}q^n_{2,.}\right)-\eta\lambda q^n_{2,.}, & \text{if } \langle u,2\rangle\in\Lambda\end{cases}\\ \ldots\ldots\\ \langle u,i\rangle\in \mathrm{K} \text{ or } \Lambda: \begin{cases} q^n_{i,.}\leftarrow q^n_{i,.}+\eta p^n_{u,.}\left(y_{u,i}-p^n_{u,.}q^n_{i,.}\right)-\eta\lambda q^n_{i,.}, & \text{if } \langle u,i\rangle\in\mathrm{K}\\ q^n_{i,.}\leftarrow q^n_{i,.}+\eta p^n_{u,.}\alpha\left(\bar{y}_{u,i}-p^n_{u,.}q^n_{i,.}\right)-\eta\lambda q^n_{i,.}, & \text{if } \langle u,i\rangle\in\Lambda\end{cases}\\ \ldots\ldots\\ \langle u,|I|\rangle\in \mathrm{K} \text{ or } \Lambda: \begin{cases} q^n_{|I|,.}\leftarrow q^n_{|I|,.}+\eta p^n_{u,.}\left(y_{u,|I|}-p^n_{u,.}q^n_{|I|,.}\right)-\eta\lambda q^n_{|I|,.}, & \text{if } \langle u,|I|\rangle\in\mathrm{K}\\ q^n_{|I|,.}\leftarrow q^n_{|I|,.}+\eta p^n_{u,.}\alpha\left(\bar{y}_{u,|I|}-p^n_{u,.}q^n_{|I|,.}\right)-\eta\lambda q^n_{|I|,.}, & \text{if } \langle u,|I|\rangle\in\Lambda\end{cases}\end{cases} \tag{11}$$

TABLE I
ALGORITHM PMLF+

```
Steps  Input: K; Output: $P_N$ and $Q_N$.
1      initializing $d$, $\lambda$, $\eta$, $T_{max}$, $\Lambda=\emptyset$,
2      initializing $M$ number of threads for parallel computing
3      for $n$=1 to $N$
4        initializing $P_n$, and $Q_n$
5        initializing $M$ number of threads for parallel computing
6        while $t\leq T_{max}$ && not converge
7        parallel training $P_n$: % with fixed $Q_n$
8          separating $\Lambda$ and K of $\bar{Y}_{n-1}$ into $M$ blocks by rows
9          for $m$=1 to $M$
10           dispatching a thread to compute
11             for $\forall\langle u,i\rangle\in$block-$m$
12               updating $p^n_{u,\cdot}$ according to (10)
13             end for
14           closing the dispatched thread
15         end for
16       parallel training $Q_n$: % with fixed $P_n$
17         separating $\Lambda$ and K into $M$ blocks by columns
18         for $m$=1 to $M$
19           dispatching a thread to compute
20             for $\forall\langle u,i\rangle\in$block-$m$
21               updating $q^n_{i,\cdot}$ according to (11)
22             end for
23           closing the dispatched thread
24         end for
25         $t$=$t$+1
26       end while
27       while generated synthetic data do not reach pre-set criterion
28         sequentially randomly selecting a missing entry $\langle u,i\rangle$ between
           two known entries in each row of $\bar{Y}_{n-1}$ as the blank entry
29         predicting $\langle u,i\rangle$ via $\hat{y}_{u,i}=\sum_{\tau=1}^{d}p^n_{u,\tau}q^n_{i,\tau}$
30         inputting $\hat{y}_{u,i}$ into (9) to obtain $\bar{y}_{u,i}$
31         putting $\bar{y}_{u,i}$ into $\Lambda$
32       end while
33     end for
```

TABLE II
DESCRIPTIONS OF STATE-OF-THE-ART COMPARISON BASELINES AND THE PROPOSED MODELS

Model	Description
BLF [8]	A basic LF model. It wins the Netflix Prize and has been extensively applied to represent HiDS data.
FNLF [47]	A fast non-negative LF model. It improves a standard non-negative LF model's representation learning ability by considering the momentum effects.
DLF [11]	A deep LF model. It has a deep structure to play a role like regularization for improving BLF's representation learning ability.
AutoRec [24]	An autoencoder-based framework [27]. It is the representative model of DNN-based model to represent HiDS data from recommender systems.
NRT [31]	A DNN-based model with a multitask learning framework [31]. It is constructed by combining recurrent neural networks and multilayer perceptron.
DCCR [27]	A DNN-based model. It improves AutoRec's representation learning ability by using two different neural networks.
PMLF [52]	The previous prediction-sampling-based multilayer- structured LF model.
PMLF+	The improved PMLF model by employing a weighting strategy and applying ASGD to achieve parallel training in this paper.

3) Baselines: To evaluate PMLF+, we compare it with six state-of-the-art baselines. They are briefly described in Table II, where three baselines are LF-based model (BLF, fast non-negative LF (FNLF), and DLF) and three baselines [AutoRec, neural rating and tips (NRT), and deep collaborative conjunctive recommender (DCCR)] are DNN-based model. Note that the previous version of PMLF+, i.e., PMLF is also compared in the experiments.

4) Implementation Details: We set $d = 10$ for PMLF+, PMLF, and the three LF-based models to draw a fair comparison. For AutoRec, NRT, and DCCR, their hidden units and layers are set according to their original papers. For PMLF and PMLF+, their settings include: 1) we increase their layer count until their prediction accuracy is not enhanced by a deeper layer or $N = 20$; 2) regarding parallel training, PMLF is implemented based on Hogwild! [59], while PMLF+ is implemented based on ASGD as analyzed in Section IV-D, and their default $M = 8$; and 3) the default $\alpha = 1.5$ for PMLF+. Besides, we tune the hyperparameters of learning rate and regularization coefficient for all the models to ensure that they can achieve their own best prediction accuracy. Each model is trained until it consumes 500 iterations or the error difference between two consecutive iterations is smaller than 10^{-6}. Besides, each model is tested five times and we report the average results. All the experiments are run on the CPU of a computer server with 2.1-GHz E5-2620, 32 cores, and 256-GB RAM.

B. Performance Comparison (RQ.1)

1) Comparison of Prediction Accuracy: The comparison results are recorded in Table III, where statistical analyses are conducted to understand these comparison RMSEs and MAEs better. First, the win/loss counts of PMLF+ versus other comparison models are presented in the second-to-last row of Table III. Second, the Friedman test [48], designed to validate the performance of multiple models on multiple datasets, is adopted to check these comparison RMSEs and MAEs. The statistical results of the Friedman test are presented in the last row of Table III. From Table III, we observe that PMLF+ wins the other comparison models in most cases, and its achieved F-rank is the lowest. Hence, these observations show that PMLF+ has the highest prediction accuracy among all the models.

Moreover, to check whether PMLF+ has a significantly lower RMSE/MAE than each comparison model, the Wilcoxon signed-ranks test [48] is also adopted to check the comparison RMSEs and MAEs in Table III [22]. There are three indicators of the Wilcoxon signed-ranks test, i.e., $R+$, $R-$, and p-value. The larger $R+$ value indicates a lower RMSE/MAE, and the p-value indicates the significance level. Table IV presents the tested results, where we find that all the hypotheses are accepted with a significance level of 0.05 except for PMLF+ versus NRT, which verifies that PMLF+ has a significantly higher prediction accuracy than BLF, FNLF, DLF, AutoRec, DCCR, and PMLF. When compared with NRT, although the hypothesis is not accepted, PMLF+ still achieves a much larger $R+$ than it does, which shows that PMLF+ has a slightly higher prediction accuracy.

2) Comparison of Computational Efficiency: Each model is tested on all the datasets to measure the CPU running time. The results are presented in Fig. 5, where we have

TABLE III
COMPARISON RESULTS ON RMSES/MAES INCLUDING FRIEDMAN TEST AND WIN/LOSS COUNTS, WHERE •DENOTES PMLF+ ACHIEVES A LOWER RMSE/MAE THAN THE OTHER MODELS

Dataset	Metric	BLF	FNLF	DLF	AutoRec	NRT	DCCR	PMLF	PMLF+
D1	RMSE	2.0616●	2.1028●	2.0502●	2.2863●	2.3033●	2.2712●	2.0271●	2.0164
	MAE	1.4269●	1.4964●	1.4258●	1.8429●	1.7948●	1.8358●	1.4093●	1.4017
D2	RMSE	1.2679●	1.2566●	1.2579●	1.2297	1.2268	1.2298	1.2416●	1.2316
	MAE	0.5960●	0.5801	0.5958●	0.5943●	0.5899●	0.5960●	0.5880●	0.5848
D3	RMSE	1.0706●	1.0895●	1.0712●	1.0603●	1.0609●	1.0630●	1.0652●	1.0586
	MAE	0.7431●	0.7555●	0.7465●	0.7325	0.7332	0.7331	0.7422●	0.7370
D4	RMSE	0.9168●	0.9278●	0.9160●	0.9187●	0.9176●	0.9169●	0.9138●	0.9131
	MAE	0.7238●	0.7271●	0.7234●	0.7285●	0.7273●	0.7265●	0.7212●	0.7208
Statistic	Win/Loss	8/0	7/1	8/0	6/2	6/2	6/2	8/0	—
	F-rank*	5.4375	5.875	4.875	5	4.75	4.9375	3.25	**1.875**

* A lower F-rank value indicates a higher rating prediction accuracy. The hypothesis that these comparison models have significant differences with a significance level of 0.05 is accepted.

TABLE IV
WILCOXON SIGNED-RANK TEST RESULTS ON RMSES/MAES OF TABLE III

Comparison	*R*+	*R*-	*p*-value*
PMLF+ vs. BLF	36	0	**0.0039**
PMLF+ vs. FNLF	35	1	**0.0078**
PMLF+ vs. DLF	36	0	**0.0039**
PMLF+ vs. AutoRec	31	5	**0.0391**
PMLF+ vs. NRT	30	6	0.0547
PMLF+ vs. DCCR	32	4	**0.0273**
PMLF+ vs. PMLF	36	0	**0.0039**

* The accepted hypotheses with a significance level of 0.05 are highlighted.

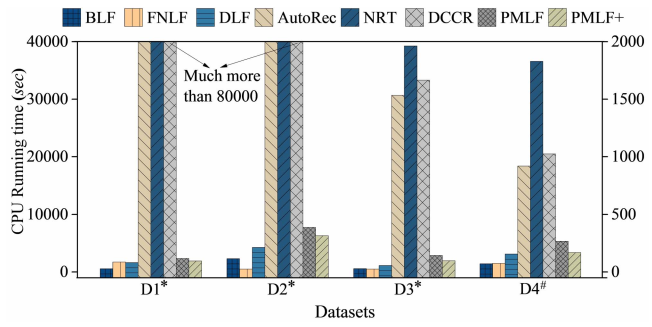

Fig. 5. CPU running time of involved models on all the datasets, where $*$ indicates that D1–D3 belongs to left tick label and # indicates that D4 belongs right tick label.

the following three findings. First, PMLF+ consumes a little more CPU running time than the three LF-based models of BLF, FNLF, and DLF. One reason is that PMLF+ trains some extra synthetic data generated by its prediction-sampling-based multilayer structure. Second, PMLF+ consumes much less CPU running time than three DNN-based models of AutoRec, NRT, and DCCR. The main reason is that they are trained on the complete data of an HiDS matrix, while PMLF+ is done only on the known ratings of an HiDS. Third, PMLF+ consumes less CPU running time than its previous version PMLF. The reason is that PMLF+ is parallelized based on ASGD, while PMLF is done based on Hogwild! [59].

C. Impact of Prediction-Sampling-Based Multilayer Structure in PMLF (RQ.2)

This set of experiments tests the changes of RMSE/MAE of PMLF+ when the layer count increases from 1 to 20. Figs. 6 and 7 present the tested results, where we observe that the following conditions hold.

1) The prediction-sampling-based multilayer structure has no impact on PMLF+'s convergence. At each layer, RMSE/MAE keeps decreasing as iteration count increases until convergence.
2) The prediction-sampling-based multilayer structure can improve an LF model's prediction accuracy. For example, at the first layer ($n = 1$), PMLF+ degenerates into a basic LF model (i.e., BLF) as there are no generated synthetic data. PMLF+ reduces the RMSE/MAE of BLF ($n = 1$) by 1.78%/1.08%, 2.77%/3.24%, 1.27%/0.76%, and 0.28%/0.07% on D1–4, respectively.
3) A deeper layer does not always make PMLF+ achieve a lower RMSE/MAE. For example, RMSE on D1 decreases from 1st layer to 5th layer. After that, RMSE increases as the layer becomes deeper. One possible reason is that after PMLF+ grows over its optimal layer threshold, some unreliable synthetic data are input into the next layer to degrade the PMLF+'s prediction accuracy.

D. Impact of Generated Synthetic Data (RQ.3)

Some researchers [1], [49], [50] have demonstrated that a multilayered cascade structure is a useful design for boosting a basic model's performance. Its basic principle is that the output of the preceding layer can be used to enrich the input

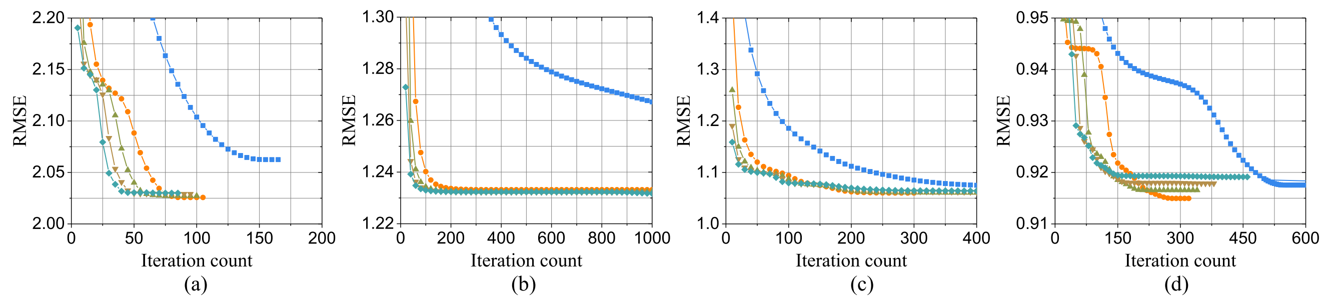

Fig. 6. Changes of RMSE of PMLF+ during the training process on all the datasets at different layers, where $\lambda = 0.01$ and $\eta = 0.001$. (a) D1. (b) D2. (c) D3. (d) D4.

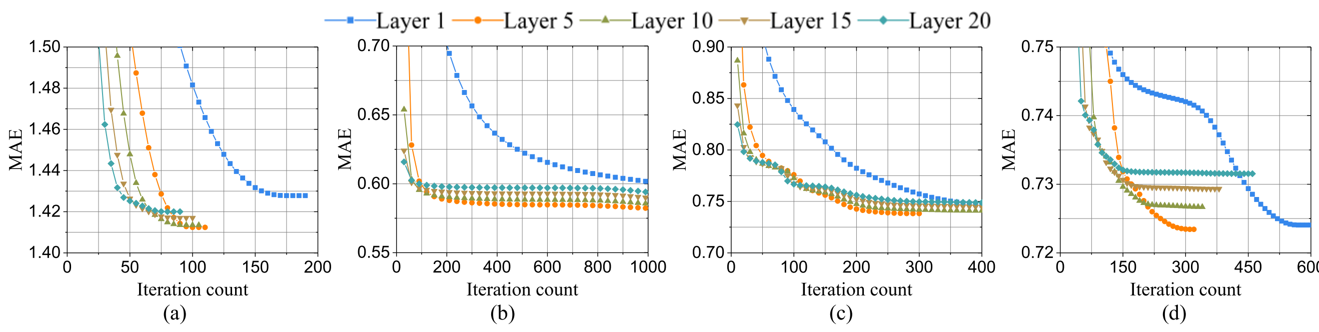

Fig. 7. Changes of MAE of PMLF+ during the training process on all the datasets at different layers, where $\lambda = 0.01$ and $\eta = 0.001$. (a) D1. (b) D2. (c) D3. (d) D4.

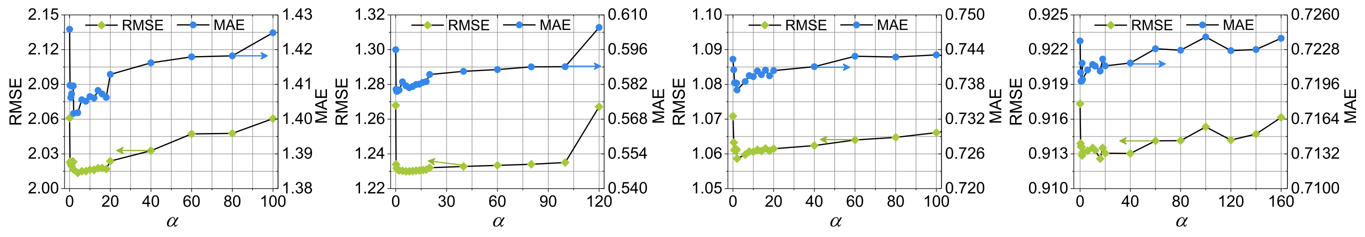

Fig. 8. Changes of RMSE/MAE of PMLF+ as α increases on all the datasets.

of the next layer. For instance, deep forest [1] constructs a multilayered cascade forest, where each layer receives the generated prediction features from its preceding layer as the additional input. With such a design, the basic model of the forest is gradually enhanced.

Following this principle, each layer of PMLF+ is designed to receive additional synthetic data generated from its preceding layer. To evaluate the effectiveness of the generated synthetic data in improving PMLF+'s representation learning ability, we test the changes of RMSE/MAE of PMLF+ as coefficient α increases. With a larger α, the generated synthetic data have more impact on PMLF+. Fig. 8 records the results on all the datasets, where we find that RMSE/MAE decreases as α increases at the beginning. After reaching a certain threshold, as α continues to increase, RMSE/MAE increases. For example, RMSE is reduced from 2.0609 to 2.0136 when α increases from 0 to 4 on D1. After that, RMSE keeps increasing to 2.0604 as α increases to 100. This finding validates that the generated synthetic data can improve PMLF+'s representation learning ability to an HiDS matrix. However, it is not necessary to make the generated synthetic data strongly impact PMLF+.

E. Impact of LF Dimension d (RQ.4)

This set of experiments tests RMSE/MAE of PMLF+ at the different layers when d increases from 10 to 160. Figs. 9 and 10 present the tested results, where we observe that a larger d makes PMLF+ achieve a lower RMSE/MAE, which verifies that a larger d makes PMLF+ possess a better representation learning ability to an HiDS matrix. However, as analyzed in Section IV-E, a larger d leads to higher time complexity. Thus, d should be appropriately set according to the specific task to balance time complexity and prediction accuracy. As a rule of thumb, d is recommended to set as 10–20.

F. Summary of Experiments

Based on the above experimental results, we summarize PMLF+'s advantages and disadvantages as follows.

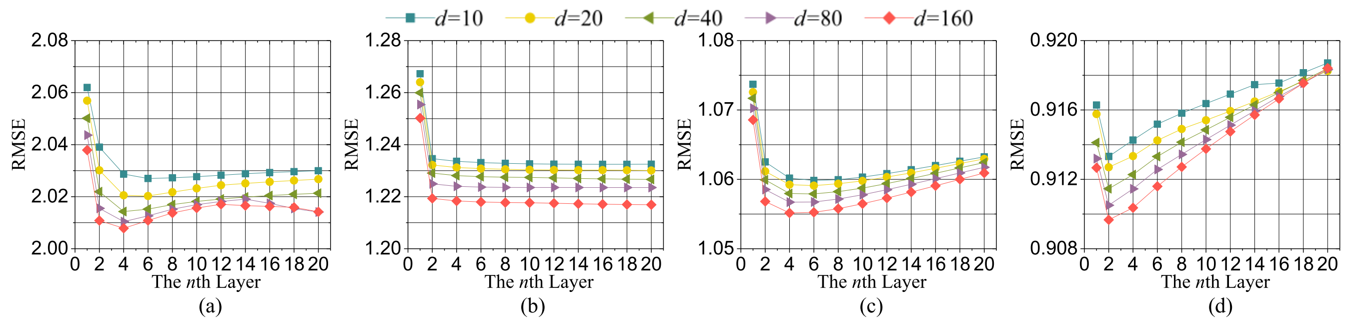

Fig. 9. Lowest RMSE as d increases from 10 to 160 on all the datasets at different layers, where $\lambda = 0.01$ and $\eta = 0.001$. (a) D1. (b) D2. (c) D3. (d) D4.

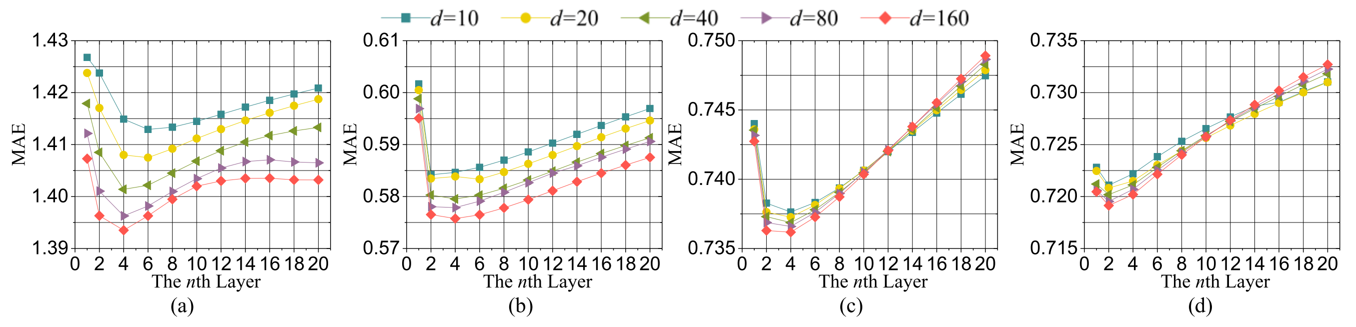

Fig. 10. Lowest MAE as d increases from 10 to 160 on all the datasets at different layers, where $\lambda = 0.01$ and $\eta = 0.001$. (a) D1. (b) D2. (c) D3. (d) D4.

1) Advantages:

1) PMLF+ can enhance a BLF model's representation learning ability by enriching its inputs via a prediction-sampling-based multilayer structure.
2) PMLF+ outperforms six state-of-the-art LF- and DNN-based models in predicting the missing data of an HiDS matrix.
3) PMLF+ improves the previous PMLF model's prediction accuracy and computational efficiency.

2) Disadvantages:

1) PMLF+ has to tune hyperparameter α to achieve the best performance. Currently, we address this issue by conducting a pretuning process on warming-up datasets. We plan to make α self-adaptive as future work.
2) Compared with LF-based models, PMLF+ needs extra computation to train the generated synthetic data. However, such extra computational costs can be significantly reduced through parallelization [40], [54].

VI. Conclusion

Following the principle of deep forest [1], this study proposes a PMLF model. PMLF has a loosely connected multilayered LF structure to generate synthetic data layer by layer, aiming at enriching the input of an LF model to enhance its representation learning ability to an HiDS matrix. To evaluate the proposed model, we conduct extensive experiments on four HiDS matrices from industrial applications. The results demonstrate that: 1) PMLF outperforms six state-of-the-art LF-based and DNNs-based models in predicting the missing data of an HiDS matrix and 2) the synthetic data generated by the prediction-sampling-based multilayer structure can improve PMLF's representation learning ability to an HiDS matrix.

Note that this study extends its previous conference version [52] to propose an improved PMLF, named PMLF+. PMLF+ has a coefficient of controlling the impact of synthetic data. Currently, this coefficient is manually tuned to guarantee better performance. Such manual tuning, however, is time-consuming and tedious. Therefore, it is crucial to make this coefficient self-adaptive. In the future, we plan to address this challenge through evolutionary computation algorithms [45], [46].

Appendix

Theorem 1: Given an HiDS matrix Y, a PMLF+ model is convergent in training two LF matrices to make its rank-d approximation Y^ by minimizing the distances between Y and Y^ only on its observed entries set K with ASGD .

Proof: From Section IV-B, we see that each layer of PMLF+ is actually individual. Hence, if the general nth layer of PMLF+ is proven to be convergent, then a PMLF+ model is convergent. To this end, we analyze the convergence of objective function (5). Since (5) is the sum of the instant loss (6), i.e., $\varepsilon(P_n,\ Q_n) = \sum_{\langle} u, i\rangle \in \Lambda \bigcup \mathrm{K}\varepsilon n\ \ u,i$ and nonconvex, some relaxations are made to make (5) converge as follows [2], [3].

Instant loss $\varepsilon_{u,i}^{n}$ is considered instead of the sum loss $\varepsilon(P_n, Q_n)$ as we adopt a single-LF-dependent form for each training, which corresponds to the single element $\forall\langle u, i\rangle \in \Lambda$ or K.

One-half of the nonconvex term is fixed to make the instant loss $\varepsilon_{u,i}^{n}$ convex, i.e., $q_{i,\cdot}^{n}$ is treated as a constant to show the convergence with the training of $p_{u,\cdot}^{n}$. Note that the training rules are symmetric for $p^n_{u,.}$ and $q^n_{i,.}$. Therefore, the convergence with the training of q^n_i, can be achieved in the same way. Next, we define the L-smooth and strong convex function $f(\xi)$ [55].

Definition 3 (L-Smooth Function $f(\xi)$): $f(\xi)$ is L-smooth if

$$\forall \xi_1, \xi_2 \in \mathrm{R}^d \text{ s.t. } \|\nabla f(\xi_1) - \nabla f(\xi_2)\|_2 \leq L\|\xi_1 - \xi_2\|_2. \tag{12}$$

Definition 4 (Strong Convex$f(\xi)$): $f(\xi)$ is strong convex if there exists a constant $\delta > 0$ satisfying

$$\forall \xi_1, \xi_2 \in \mathrm{R}^d \text{s.t.} f(\xi_1) \geq f(\xi_2) + \nabla f(\xi_2)(\xi_1 - \xi_2)^{\mathrm{T}} + \frac{1}{2}\delta\|\xi_1 - \xi_2\|_2^2. \tag{13}$$

Lemma 1: The instant loss $\varepsilon^n_{u,i}$ is L-smooth when L is the maximum singular value for matrix $(q^{n\,\mathrm{T}}_{i,.}q^n_{i,.} + \lambda E_d)$ and E_d is a $d \times d$ identity matrix.

Proof: Assuming that $p_{\sigma,}.$ and $p_{\varphi,}$ are two arbitrary and independent row vectors of P_n, we have (14), as shown at the bottom of the page.

Note that to simply (14) for convergence analysis, we can make $\alpha = 1$. Then, with (14), we achieve that

$$\begin{aligned} &\left\|\nabla\varepsilon^n_{u,i}(p_{\sigma,.}) - \nabla\varepsilon^n_{u,i}(p_{\varphi,.})\right\|_2 \\ &\quad= \left\|(p_{\sigma,.} - p_{\varphi,.})\left(q^{n\,\mathrm{T}}_{i,.}q^n_{i,.} + \lambda E_d\right)\right\|_2. \end{aligned} \tag{15}$$

According to the L_2-norm properties of a matrix [56], we have the following inequality:

$$\begin{aligned} &\left\|\nabla\varepsilon^n_{u,i}(p_{\sigma,.}) - \nabla\varepsilon^n_{u,i}(p_{\varphi,.})\right\|_2 \\ &\quad\leq \left\|\left(q^{n\,\mathrm{T}}_{i,.}q^n_{i,.} + \lambda E_d\right)\right\|_2 \left\|(p_{\sigma,.} - p_{\varphi,.})\right\|_2 \end{aligned} \tag{16}$$

where $||(q^{n\,\mathrm{T}}_{i,.}q^n_{i,.} + \lambda E_d)||_2$ denotes the largest singular value of $(q^{n\,\mathrm{T}}_{i,.}q^n_{i,.} + \lambda E_d)$. Based on the above inferences, we obtain $L = ||q^{n\,\mathrm{T}}_{i,.}q^n_{i,.} + \lambda E_d)||_2$. Hence, Lemma 1 holds. □

Lemma 2: The instant loss $\varepsilon^n_{u,i}$ is of strong convexity when δ is the minimum singular value for matrix $(q^{n\,\mathrm{T}}_{i,.}q^n_{i,.} + \lambda E_d)$.

Proof: Given arbitrary vectors $p_{\sigma,.}$ and $p_{\varphi,.}$, we expand the state of $\varepsilon^n_{u,i}$ at $p_{\varphi,.}$, Following the principle of Taylor series given as follows:

$$\begin{aligned} &\varepsilon^n_{u,i}(p_{\sigma,.}) \\ &\quad\approx \varepsilon^n_{u,i}(p_{\varphi,.}) + \nabla\varepsilon^n_{u,i}(p_{\varphi,.})(p_{\sigma,.} - p_{\varphi,.})^{\mathrm{T}} \\ &\qquad+ \frac{1}{2}(p_{\sigma,.} - p_{\varphi,.})\nabla^2\varepsilon^n_{u,i}(p_{\varphi,.})(p_{\sigma,.} - p_{\varphi,.})^{\mathrm{T}}. \\ &\quad\Rightarrow \varepsilon^n_{u,i}(p_{\sigma,.}) - \varepsilon^n_{u,i}(p_{\varphi,.}) = \nabla\varepsilon^n_{u,i}(p_{\varphi,.})(p_{\sigma,.} - p_{\varphi,.})^{\mathrm{T}} \\ &\qquad+ \frac{1}{2}(p_{\sigma,.} - p_{\varphi,.})\nabla^2\varepsilon^n_{u,i}(p_{\varphi,.})(p_{\sigma,.} - p_{\varphi,.})^{\mathrm{T}}. \end{aligned} \tag{17}$$

As shown in Definition 4, if $\varepsilon^n_{u,i}$ is strong convex, we can achieve that

$$\begin{aligned} \varepsilon^n_{u,i}(p_{\sigma,.}) - \varepsilon^n_{u,i}(p_{\varphi,.}) \geq{}& \nabla\varepsilon^n_{u,i}(p_{\varphi,.})(p_{\sigma,.} - p_{\varphi,.})^{\mathrm{T}} \\ &+ \frac{1}{2}\delta\left\|p_{\sigma,.} - p_{\varphi,.}\right\|_2^2. \end{aligned} \tag{18}$$

Thus, Lemma 2 is equivalent to select δ to make the following inequality true:

$$(p_{\sigma,.} - p_{\varphi,.})\nabla^2\varepsilon^n_{u,i}(p_{\varphi,.})(p_{\sigma,.} - p_{\varphi,.})^{\mathrm{T}} \geq \delta\left\|p_{\sigma,.} - p_{\varphi,.}\right\|_2^2. \tag{19}$$

From the expression of $\varepsilon^n_{u,i}$, we can obtain that

$$\nabla^2\varepsilon^n_{u,i}(p_{\varphi,.}) = q^{n\,\mathrm{T}}_{i,.}q^n_{i,.} + \lambda E_d. \tag{20}$$

By combining (19) and (20), we only need to prove that

$$\begin{aligned} &(p_{\sigma,.} - p_{\varphi,.})\left(q^{n\,\mathrm{T}}_{i,.}q^n_{i,.} + \lambda E_d\right)(p_{\sigma,.} - p_{\varphi,.})^{\mathrm{T}} \\ &\quad\geq \delta\left\|(p_{\sigma,.} - p_{\varphi,.})\right\|_2^2. \end{aligned} \tag{21}$$

Furthermore, (21) is also equivalent to

$$(p_{\sigma,.} - p_{\varphi,.})(q^{\mathrm{T}}q^n_{i,.} + \lambda E_d - \delta E_d)(p_{\sigma,.} - p_{\varphi,.})^{\mathrm{T}} \geq 0. \tag{22}$$

According to the properties of the matrix, (22) is equivalent to prove that $(q^{n\,\mathrm{T}}_{i,.}q^n_{i,.} + \lambda E_d - \delta E_d)$ is a positive semidefinite matrix. As unveiled by Zhang [56], $(q^{n\,\mathrm{T}}_{i,.}q^n_{i,.} + \lambda E_d - \delta E_d)$ is a positive semidefinite matrix when δ is the minimum singular value of matrix $(q^{n\,\mathrm{T}}_{i,.}q^n_{i,.} + \lambda E_d)$, and it satisfies positive semidefiniteness. Hence, Lemma 2 holds.

Considering the tth iteration of PLFM+, we have the following training rule for $p^n_{u,.}$ on a single entry $\forall\langle u, i\rangle \in \Lambda$ or K:

$$p^n_{u,.}(\omega) \leftarrow p^n_{u,.}(\omega - 1) - \eta \cdot \nabla\varepsilon^n_{u,i}\left(p^n_{u,.}(\omega - 1)\right) \tag{23}$$

where $p^n_{u,.}(\omega)$ and $p^n_{u,.}(\omega{-}1)$ denote the state of $p^n_{u,.}$ trained by the ωth and $(\omega - 1)$th entries in the tth iteration, respectively. If $p^{n,*}_{u,.}$ is the optimal state of $p^n_{u,.}$, we have

$$\begin{aligned} &\left\|p^n_{u,.}(\omega) - p^{n,*}_{u,.}\right\|_2^2 \\ &\quad= \left\|p^n_{u,.}(\omega - 1) - \eta\nabla\varepsilon^n_{u,i}\left(p^n_{u,.}(\omega - 1)\right) - p^{n,*}_{u,.}\right\|_2^2 \\ &\quad= \left\|p^n_{u,.}(\omega - 1) - p^{n,*}_{u,.}\right\|_2^2 \\ &\qquad- 2\eta\nabla\varepsilon^n_{u,i}\left(p^n_{u,.}(\omega - 1)\right)\left(p^n_{u,.}(\omega - 1) - p^{n,*}_{u,.}\right)^{\mathrm{T}} \\ &\qquad+ \eta^2\left\|\nabla\varepsilon^n_{u,i}\left(p^n_{u,.}(\omega - 1)\right)\right\|_2^2. \end{aligned} \tag{24}$$

$$\left\{\begin{array}{l} \text{if}\langle u, i\rangle \in S_K: \left\{\begin{array}{l} \nabla\varepsilon^n_{u,i}(p_{\sigma,.}) - \nabla\varepsilon^n_{u,i}(p_{\varphi,.}) \\ = -(y_{u,i} - p_{\sigma,.}q^n_{i,.})q^n_{i,.} + \lambda p_{\sigma,.} + (y_{u,i} - p_{\varphi,.}q^n_{i,.})q^n_{i,.} - \lambda p_{\varphi,.} \\ = (p_{\sigma,.} - p_{\varphi,.})\left(q^{n\mathrm{T}}_{i,.}q^n_{i,.} + \lambda E_d\right) \end{array}\right. \\ \text{if}\langle u, i\rangle \in \Lambda: \left\{\begin{array}{l} \nabla\varepsilon^n_{u,i}(p_{\sigma,.}) - \nabla\varepsilon^n_{u,i}(p_{\varphi,.}) \\ = -\alpha(\bar{y}_{u,i} - p_{\sigma,.}q^n_{i,.})q^n_{i,.} + \lambda p_{\sigma,.} + \alpha(\bar{y}_{u,i} - p_{\varphi,.}q^n_{i,.})q^n_{i,.} - \lambda p_{\varphi,.} \\ = (p_{\sigma,.} - p_{\varphi,.})(\alpha q^{n\,\mathrm{T}}_{i,.}q^n_{i,.} + \lambda E_d) \end{array}\right. \end{array}\right. \tag{14}$$

Based on Lemma 2, we achieve that

$$\begin{aligned}\varepsilon_{u,i}^{n}\left(p_{u,\cdot}^{n,*}\right)-\varepsilon_{u,i}^{n}\left(p_{u,\cdot}^{n}(\omega-1)\right)&\\ \geq \nabla\varepsilon_{u,i}^{n}\left(p_{u,\cdot}^{n}(\omega-1)\right)\left(p_{u,\cdot}^{n,*}-p_{u,\cdot}^{n}(\omega-1)\right)^{\mathrm{T}}&\\ +\frac{1}{2}\delta\left\|p_{u,\cdot}^{n,*}-p_{u,\cdot}^{n}(\omega-1)\right\|_2^2.&\end{aligned} \tag{25}$$

Since $p_{u,\cdot}^{n,*}$ is the optimal state of $p_{u,\cdot}^{n}$, we have that

$$\begin{cases}\nabla\varepsilon_{u,i}^{n}\left(p_{u,\cdot}^{n,*}\right)=0,\\ \varepsilon_{u,i}^{n}\left(p_{u,\cdot}^{n,*}\right)<\varepsilon_{u,i}^{n}\left(p_{u,\cdot}^{n}(\omega-1)\right).\end{cases} \tag{26}$$

By substituting (26) into (25), we see that

$$\begin{aligned}\nabla\varepsilon_{u,i}^{n}\left(p_{u,\cdot}^{n}(\omega-1)\right)\left(p_{u,\cdot}^{n}(\omega-1)-p_{u,\cdot}^{n,*}\right)^{\mathrm{T}}&\\ \geq\frac{1}{2}\delta\left\|p_{u,\cdot}^{n}(\omega-1)-p_{u,\cdot}^{n,*}\right\|_2^2.&\end{aligned} \tag{27}$$

Thus, based on (27), (24) is equivalent to

$$\begin{aligned}&\left\|p_{u,\cdot}^{n}(\omega)-p_{u,\cdot}^{n,*}\right\|_2^2\\ &\leq(1-\eta\delta)\left\|p_{u,\cdot}^{n}(\omega-1)-p_{u,\cdot}^{n,*}\right\|_2^2\\ &\quad+\eta^2\left\|\nabla\varepsilon_{u,i}^{n}\left(p_{u,\cdot}^{n}(\omega-1)\right)\right\|_2^2.\end{aligned} \tag{28}$$

By taking the expectation of (28), we have

$$\begin{aligned}&\mathrm{E}\left[\left\|p_{u,\cdot}^{n}(\omega)-p_{u,\cdot}^{n,*}\right\|_2^2\right]\\ &\leq(1-\eta\delta)\mathrm{E}\left[\left\|p_{u,\cdot}^{n}(\omega-1)-p_{u,\cdot}^{n,*}\right\|_2^2\right]\\ &\quad+\eta^2\mathrm{E}\left[\left\|\nabla\varepsilon_{u,i}^{n}\left(p_{u,\cdot}^{n}(\omega-1)\right)\right\|_2^2\right].\end{aligned} \tag{29}$$

Following [57], assume that there is a positive number z such that

$$\mathrm{E}\left[\left\|\nabla\varepsilon_{u,i}^{n}\left(p_{u,\cdot}^{n}(\omega-1)\right)\right\|_2^2\right]\leq z^2. \tag{30}$$

Thus, based on (30), (29) is equivalent to

$$\begin{aligned}&\mathrm{E}\left[\left\|p_{u,\cdot}^{n}(\omega)-p_{u,\cdot}^{n,*}\right\|_2^2\right]\\ &\leq(1-\eta\delta)\mathrm{E}\left[\left\|p_{u,\cdot}^{n}(\omega-1)-p_{u,\cdot}^{n,*}\right\|_2^2\right]+\eta^2z^2.\end{aligned} \tag{31}$$

Let us take the learning rate $\eta=\beta/(\delta t)$ with $\beta>1$, (31) can be reformulated as

$$\begin{aligned}&\mathrm{E}\left[\left\|p_{u,\cdot}^{n}(\omega)-p_{u,\cdot}^{n,*}\right\|_2^2\right]\\ &\leq\left(1-\frac{\beta}{t}\right)\mathrm{E}\left[\left\|p_{u,\cdot}^{n}(\omega-1)-p_{u,\cdot}^{n,*}\right\|_2^2\right]+\frac{1}{t^2}\left(\frac{\beta z}{\delta}\right)^2.\end{aligned} \tag{32}$$

By expanding the expression of (32) by induction, we obtain a bound

$$\mathrm{E}\left[\left\|p_{u,\cdot}^{n}(\omega)-p_{u,\cdot}^{n,*}\right\|_2^2\right]\leq\frac{1}{t}\max\left\{\left\|p_{u,\cdot}^{n}(1)-p_{u,\cdot}^{n,*}\right\|_2^2,\frac{\beta^2z^2}{\delta\beta-1}\right\} \tag{33}$$

where $p_{u,\cdot}^{n}$ (1) denotes the initial state of $p_{u,\cdot}^{n}$ at the tth iteration.

According to Lemma 1, $\varepsilon_{u,i}^{n}$ is L-smooth and we can achieve that

$$\varepsilon_{u,i}^{n}\left(p_{u,\cdot}^{n}(\omega)\right)-\varepsilon_{u,i}^{n}\left(p_{u,\cdot}^{n,*}\right)\leq\frac{L}{2}\left\|p_{u,\cdot}^{n}(\omega)-p_{u,\cdot}^{n,*}\right\|_2^2. \tag{34}$$

By taking the expectation of (34), we can obtain that

$$\mathrm{E}\left[\varepsilon_{u,i}^{n}\left(p_{u,\cdot}^{n}(\omega)\right)-\varepsilon_{u,i}^{n}\left(p_{u,\cdot}^{n,*}\right)\right]\leq\frac{L}{2}\mathrm{E}\left[\left\|p_{u,\cdot}^{n}(\omega)-p_{u,\cdot}^{n,*}\right\|_2^2\right]. \tag{35}$$

By substituting (33) into (35), we have the following deduction:

$$\mathrm{E}\left[\varepsilon_{u,i}^{n}\left(p_{u,\cdot}^{n}(\omega)\right)-\varepsilon_{u,i}^{n}\left(p_{u,\cdot}^{n,*}\right)\right]\leq\frac{L}{2t}\Theta(\beta) \tag{36}$$

where we have

$$\Theta(\beta)=\max\left\{\left\|p_{u,\cdot}^{n}(1)-p_{u,\cdot}^{n,*}\right\|_2^2,\frac{\beta^2z^2}{\delta\beta-1}\right\}. \tag{37}$$

Then, we expand (36) on all the known entries of Λ and K

$$\begin{aligned}&\mathrm{E}\left[\sum_{\langle u,i\rangle\in\Lambda\cup S_K}\left(\varepsilon_{u,i}^{n}\left(p_{u,\cdot}^{n}(\omega)\right)-\varepsilon_{u,i}^{n}\left(p_{u,\cdot}^{n,*}\right)\right)\right]\\ &\qquad\leq(|\Lambda|+|S_K|)\frac{L}{2t}\Theta(\beta)\end{aligned} \tag{38}$$

where $t\to\infty$, and we have $(|\Lambda|+|\mathrm{K}|)(L/2t)\Theta(\beta)\to 0$.

We can encounter the same situation when $p_{u,\cdot}^{n}$ is treated as a constant. Although the learning objective (5) is nonconvex, $p_{u,\cdot}^{n}$ and $p_{i,\cdot}^{n}$ can be trained alternatively by ASGD. Moreover, as unveiled by [58], SGD requires the learning rate $\eta\leq 1/\delta t$ in the tth iteration. Thus, following Lemma 2, the learning rate in the tth iteration satisfies $\eta\leq 1/\delta t$, where δ is the minimum singular value of the matrix $(q_{i,\cdot}^{n\mathrm{T}}q_{i,\cdot}^{n}+\lambda E_d)$. Besides, we see that the regularization does not affect the convergence, which means that (5) is convergent. Hence, Theorem 1 holds.

References

[1] Z.-H. Zhou and J. Feng, "Deep forest: Towards an alternative to deep neural networks," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 3553–3559.

[2] X. Luo, M. Zhou, S. Li, Z. You, Y. Xia, and Q. Zhu, "A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 3, pp. 579–592, May 2015.

[3] Y. Koren and R. Bell, "Advances in collaborative filtering," in *Recommender Systems Handbook*. Cham, Switzerland: Springer, 2015, pp. 77–118.

[4] B. Smith and G. Linden, "Two decades of recommender systems at Amazon.Com," *IEEE Internet Comput.*, vol. 21, no. 3, pp. 12–18, May 2017.

[5] J. Han *et al.*, "Adaptive deep modeling of users and items using side information for recommendation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 3, pp. 737–748, Jun. 2019.

[6] J. Castro, J. Lu, G. Zhang, Y. Dong, and L. Martinez, "Opinion dynamics-based group recommender systems," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 48, no. 12, pp. 2394–2406, Dec. 2018.

[7] X. He, J. Tang, X. Du, R. Hong, T. Ren, and T. Chua, "Fast matrix factorization with nonuniform weights on missing data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 8, pp. 2791–2804, Aug. 2020.

[8] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Comput.*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[9] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "QoS-aware web service recommendation by collaborative filtering," *IEEE Trans. Services Comput.*, vol. 4, no. 2, pp. 140–152, Apr./Jun. 2011.

[10] M. Pang, K.-M. Ting, P. Zhao, and Z.-H. Zhou, "Improving deep forest by confidence screening," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2018, pp. 1194–1199.

[11] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, and M. Zhou, "A deep latent factor model for high-dimensional and sparse matrices in recommender systems," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 51, no. 7, pp. 4285–4296, Jul. 2021.

[12] H. Wu, Z. Zhang, K. Yue, B. Zhang, J. He, and L. Sun, “Dual-regularized matrix factorization with deep neural networks for recommender systems,” *Knowl.-Based Syst.*, vol. 145, pp. 46–58, Apr. 2018.

[13] X. Ren, M. Song, E. Haihong, and J. Song, “Context-aware probabilistic matrix factorization modeling for point-of-interest recommendation,” *Neurocomputing*, vol. 241, pp. 38–55, Jun. 2017.

[14] H. Liu, L. Jing, J. Yu, and M. K. Ng, “Social recommendation with learning personal and social latent factors,” *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 7, pp. 2956–2970, Jul. 2021.

[15] D. Ryu, K. Lee, and J. Baik, “Location-based web service QoS prediction via preference propagation to address cold start problem,” *IEEE Trans. Services Comput.*, vol. 14, no. 3, pp. 736–746, May 2021.

[16] C. Leng, H. Zhang, G. Cai, I. Cheng, and A. Basu, “Graph regularized Lp smooth non-negative matrix factorization for data representation,” *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 2, pp. 584–595, Mar. 2019.

[17] C. Wang *et al.*, “Confidence-aware matrix factorization for recommender systems,” in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 434–442.

[18] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, p. 436, Feb. 2015.

[19] S. Zhang, L. Yao, A. Sun, and Y. Tay, “Deep learning based recommender system: A survey and new perspectives,” *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–38, Jan. 2020.

[20] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, “Neural collaborative filtering,” in *Proc. 26th Int. Conf. World Wide Web (WWW)*, vol. 2017, pp. 173–182.

[21] H.-J. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, “Deep matrix factorization models for recommender systems,” in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 3203–3209.

[22] S. Zhang, Y. Tay, L. Yao, B. Wu, and A. Sun, “DeepRec: An open-source toolkit for deep learning based recommendation,” in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 6581–6583.

[23] S. Zhang, L. Yao, A. Sun, S. Wang, G. Long, and M. Dong, “NeuRec: On nonlinear transformation for personalized ranking,” in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 3669–3675.

[24] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, “AutoRec: Autoencoders meet collaborative filtering,” in *Proc. 24th Int. Conf. World Wide Web*, May 2015, pp. 111–112.

[25] S. Li, J. Kawale, and Y. Fu, “Deep collaborative filtering via marginalized denoising auto-encoder,” in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2015, pp. 811–820.

[26] S. Cao, N. Yang, and Z. Liu, “Online news recommender based on stacked auto-encoder,” in *Proc. IEEE/ACIS 16th Int. Conf. Comput. Inf. Sci. (ICIS)*, May 2017, pp. 721–726.

[27] Q. Wang, B. Peng, X. Shi, T. Shang, and M. Shang, “DCCR: Deep collaborative conjunctive recommender for rating prediction,” *IEEE Access*, vol. 7, pp. 60186–60198, 2019.

[28] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, “Collaborative denoising auto-encoders for top-N recommender systems,” in *Proc. 9th ACM Int. Conf. Web Search Data Mining*, Feb. 2016, pp. 153–162.

[29] S. Okura, Y. Tagami, S. Ono, and A. Tajima, “Embedding-based news recommendation for millions of users,” in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 1933–1942.

[30] X. Dong, L. Yu, Z. Wu, Y. Sun, L. Yuan, and F. Zhang, “A hybrid collaborative filtering model with deep structure for recommender systems,” in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 1309–1315.

[31] P. Li, Z. Wang, Z. Ren, L. Bing, and W. Lam, “Neural rating regression with abstractive tips generation for recommendation,” in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 2017, pp. 345–354.

[32] X. He and T.-S. Chua, “Neural factorization machines for sparse predictive analytics,” in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 2017, pp. 355–364.

[33] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T.-S. Chua, “Attentional factorization machines: Learning the weight of feature interactions via attention networks,” in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 3119–3125.

[34] L. Zheng, V. Noroozi, and P. S. Yu, “Joint deep modeling of users and items using reviews for recommendation,” in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, Feb. 2017, pp. 425–434.

[35] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, “Convolutional matrix factorization for document context-aware recommendation,” in *Proc. 10th ACM Conf. Recommender Syst.*, Sep. 2016, pp. 233–240.

[36] P. Massa and P. Avesani, “Trust-aware recommender systems,” in *Proc. ACM Conf. Recommender Syst. (RecSys)*, 2007, pp. 17–24.

[37] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, and X. Wu, “A data-characteristic-aware latent factor model for web services QoS prediction,” *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 6, pp. 2525–2538, Jun. 2022.

[38] H. Li, K. Li, J. An, and K. Li, “MSGD: A novel matrix factorization approach for large-scale collaborative filtering recommender systems on GPUs,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 7, pp. 1530–1544, Jul. 2017.

[39] H. Li, K. Li, J. An, W. Zheng, and K. Li, “An efficient manifold regularized sparse non-negative matrix factorization model for large-scale recommender systems on GPUs,” *Inf. Sci.*, vol. 496, pp. 464–484, Sep. 2019.

[40] X. Luo, H. Liu, G. Gou, Y. Xia, and Q. Zhu, “A parallel matrix factorization based recommender by alternating stochastic gradient decent,” *Eng. Appl. Artif. Intell.*, vol. 25, no. 7, pp. 1403–1412, 2012.

[41] X. Luo, Z. Wang, and M. Shang, “An instance-frequency-weighted regularization scheme for non-negative latent factor analysis on high-dimensional and sparse data,” *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 51, no. 6, pp. 3522–3532, Jun. 2021.

[42] L. Brozovsky and V. Petricek, “Recommender system for online dating service,” 2007, *arXiv:cs/0703042*.

[43] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, “Eigentaste: A constant time collaborative filtering algorithm,” *Inf. Retr.*, vol. 4, no. 2, pp. 133–151, Jul. 2001.

[44] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, “GroupLens: Applying collaborative filtering to UseNet news,” *Commun. ACM*, vol. 40, no. 3, pp. 77–87, 1997.

[45] Bilal, M. Pant, H. Zaheer, L. Garcia-Hernandez, and A. Abraham, “Differential evolution: A review of more than two decades of research,” *Eng. Appl. Artif. Intell.*, vol. 90, Apr. 2020, Art. no. 103479.

[46] Y. Wang, S. Gao, M. Zhou, and Y. Yu, “A multi-layered gravitational search algorithm for function optimization and real-world problems,” *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 1, pp. 94–109, Jan. 2021.

[47] X. Luo, Z. Liu, S. Li, M. Shang, and Z. Wang, “A fast non-negative latent factor model based on generalized momentum method,” *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 1, pp. 610–620, Jan. 2021.

[48] J. Demšar, “Statistical comparisons of classifiers over multiple data sets,” *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Dec. 2006.

[49] J. Gama and P. Brazdil, “Cascade generalization,” *Mach. Learn.*, vol. 41, no. 3, pp. 315–343, Dec. 2000.

[50] H. Zhao and S. Ram, “Constrained cascade generalization of decision trees,” *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 6, pp. 727–739, Jun. 2004.

[51] X. Mao, K. Qiu, T. Li, and Y. Gu, “Spatio-temporal signal recovery based on low rank and differential smoothness,” *IEEE Trans. Signal Process.*, vol. 66, no. 23, pp. 6281–6296, Dec. 2018.

[52] D. Wu, L. Jin, and X. Luo, “PMLF: Prediction-sampling-based multilayer-structured latent factor analysis,” in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2020, pp. 671–680.

[53] S. Rendle, W. Krichene, L. Zhang, and J. Anderson, “Neural collaborative filtering vs. matrix factorization revisited,” in *Proc. 14th ACM Conf. Recommender Syst.*, Sep. 2020, pp. 240–248.

[54] X. Shi, Q. He, X. Luo, Y. Bai, and M. Shang, “Large-scale and scalable latent factor analysis via distributed alternative stochastic gradient descent for recommender systems,” *IEEE Trans. Big Data*, vol. 8, no. 2, pp. 420–431, Apr. 2022.

[55] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2009.

[56] X. D. Zhang, *Matrix Analysis and Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2017.

[57] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, “Robust stochastic approximation approach to stochastic programming,” *SIAM J. Optim.*, vol. 19, no. 4, pp. 1574–1609, 2009.

[58] A. Rakhlin, O. Shamir, and K. Sridharan, “Making gradient descent optimal for strongly convex stochastic optimization,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 2012, pp. 1571–1578.

[59] B. Recht, C. Re, S. Wright, and F. Niu, “Hogwild: A lock-free approach to parallelizing stochastic gradient descent,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 693–701.

[60] X. Luo, W. Qin, A. Dong, K. Sedraoui, and M. Zhou, “Efficient and high-quality recommendations via momentum-incorporated parallel stochastic gradient descent-based learning,” *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 2, pp. 402–411, Feb. 2021.

[61] X. Luo, D. Wang, M. Zhou, and H. Yuan, “Latent factor-based recommenders relying on extended stochastic gradient descent algorithms,” *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 51, no. 2, pp. 916–926, Feb. 2021.

[62] W. Peng, L. Li, W. Dai, J. Du, and W. Lan, "Predicting protein functions through non-negative matrix factorization regularized by protein-protein interaction network and gene functional information," in *Proc. IEEE Int. Conf. Bioinf. Biomed. (BIBM)*, Nov. 2019, pp. 86–89.

[63] Y. Wang, Y. Zhang, L. Wang, Y. Hu, and B. Yin, "Urban traffic pattern analysis and applications based on spatio-temporal non-negative matrix factorization," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 12752–12765, Aug. 2022, doi: 10.1109/TITS.2021.3117130.

[64] B. Hosseini and B. Hammer, "Interpretable discriminative dimensionality reduction and feature selection on the manifold," in *Proc. Eur. Conf. Mach. Learn. Knowl. Discovery Database (ECML PKDD)*, vol. 2019, pp. 310–326.

[65] B. Hosseini and B. Hammer, "Large-margin multiple kernel learning for discriminative features selection and representation learning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–8.

[66] B. Hosseini and B. Hammer, "Confident kernel sparse coding and dictionary learning," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2018, pp. 1031–1036.

[67] D. Wu, Y. He, X. Luo, and M. Zhou, "A latent factor analysis-based approach to online sparse streaming feature selection," *IEEE Trans. Syst., Man, Cybern. Syst.*, early access, Aug. 4, 2021, doi: 10.1109/TSMC.2021.3096065.

[68] J. Bi, H. Yuan, J. Zhai, M. Zhou, and H. V. Poor, "Self-adaptive bat algorithm with genetic operation," *IEEE/CAA J. Autom. Sinica*, vol. 9, no. 7, pp. 1284–1294, Jul. 2022.

[69] Y. Hua, Q. Liu, K. Hao, and Y. Jin, "A survey of evolutionary algorithms for multi-objective optimization problems with irregular Pareto fronts," *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 2, pp. 303–318, Feb. 2021.

Di Wu (Member, IEEE) received the Ph.D. degree from the Chongqing Institute of Green and Intelligent Technology (CIGIT), Chinese Academy of Sciences (CAS), Chongqing, China, in 2019.

He joined CIGIT, CAS. He is currently a Professor with the College of Computer and Information Science, Southwest University, Chongqing. He has authored or coauthored over 50 publications, including journals of IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, TKDE, TSMC, and TSC, and conferences of ICDM, WWW, and IJCAI. His current research interests include machine learning and data mining. His homepage: https://wuziqiao.github.io/Homepage/

Xin Luo (Senior Member, IEEE) received the B.S. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2005, and the Ph.D. degree in computer science from Beihang University, Beijing, China, in 2011.

He is currently a Professor of data science and computational intelligence with the College of Computer and Information Science, Southwest University, Chongqing, China. His current research interests include data science. He has authored or coauthored over 200 papers (including over 90 IEEE TRANSACTIONS papers) in the areas of his interests.

Dr. Luo was a recipient of the Outstanding Associate Editor Award from IEEE/CAA JOURNAL OF AUTOMATICA SINICA in 2020. He is also an Associate Editor of IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS and IEEE/CAA JOURNAL OF AUTOMATICA SINICA.

Yi He (Member, IEEE) received the Ph.D. degree from the University of Louisiana at Lafayette, Lafayette, LA, USA, in 2020.

He joined Old Dominion University, Norfolk, VA, USA, where he is currently an Assistant Professor. His research outcomes have appeared in premier venues, e.g., AAAI, IJCAL, WWW, ICDM, TKDET, and TNNLS, among many other Al, machine learning, and data mining outlets. His homepage: https://www.lions.odu.edu/∼y1he/index.html

MengChu Zhou (Fellow, IEEE) received the B.S. degree from the Nanjing University of Science and Technology, Nanjing, China, in 1983, the M.S. degree from the Beijing Institute of Technology, Beijing, China, in 1986, and the Ph.D. degree from the Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990.

In 1990, he joined the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA, where he is currently a Distinguished Professor. He has more than 1000 publications including 12 books, more than 700 journal articles (more than 600 IEEE TRANSACTIONS articles), 31 patents, and 30 book-chapters. His research interests include intelligent automation, Petri nets, the Internet of Things, edge/cloud computing, and big data analytics.

Dr. Zhou is a fellow ofthe International Federation of Automatic Control (IFAC), American Association for the Advancement of Science (AAAS), Chinese Association of Automation, and the National Academy of Inventors. He was a recipient of the Excellence in Research Prize and Medal from NJIT, the Humboldt Research Award for U.S. Senior Scientists from Alexander von Humboldt Foundation, the Franklin V. Taylor Memorial Award and the Norbert Wiener Award from the IEEE Systems, Man, and Cybernetics Society, and the Edison Patent Award from the Research and Development Council of New Jersey.